Geoff Huston
Joao Luis Silva Damas
May 2016

DNS Privacy

The DNS is normally a relatively open protocol that smears its data (which is your data and mine too!) far and wide. Little wonder that the DNS is used in many ways, not just as a mundane name resolution protocol, but as a data channel for surveillance and as a common means of implementing various forms of content access control. But all this is poised to change. Now that the Snowden files have sensitized us to the level of such activities, we have become acutely aware that many of our tools are just way too trusting, way too chatty, and way too easily subverted. First and foremost in this collection of vulnerable tools is the Domain Name System.

Queries made to the Domain Name System are a precursor to almost every Internet transaction. Obtaining a log of the DNS queries I make is perhaps the equivalent in terms of information content to obtaining a telephone's log of called numbers from a previous generation. A DNS transaction log may not provide information about the precise network transactions I made, but it does record which sites I've been using, which often is not just good enough, its exactly what someone needs in order to build a highly accurate profile of what I do on the Internet. It's not just national security bodies that have such an interest. These days we see many systems that target the individual user, and build a comprehensive profile of their needs and desires. The difference between an annoying advertisement and a timely helpful suggestion is just information about the user, and many companies assemble such profiles as part of their own commercial activities (Figure 1).

The DNS is incredibly chatty. For example, to resolve a new name, such as `www.example.com`, a DNS resolver would first ask the root name servers for the IP address of `www.example.com`. The root name servers would not be able to provide the answer, but they will respond with the authoritative name servers for the `.com` domain. The resolver will then repeat this query relating to the IP address of the name `www.example.com.` to a `.com` name server, and once more the answer is an indirect one, indicating that while it does not know the answer, the list of name server for the domain `example.com` should be queried. At this point the resolver can repeat the same query to a server that is authoritative for the `example.com` domain and probably receive an answer that contains the address of `www.example.com`. But let's think about these queries for a second. In this case a root server, a `.com` server and an `example.com` server are all aware that I am "interested" in `www.example.com.`, and they probably have stored a log of these queries. I have no idea if these logs are private or public. I have no idea how they get analyzed, and what inferences are drawn from this data.

It's possible it is a little worse than this, as the application I am using, such as a browser normally does not perform DNS name resolution itself. It passes the query to the platform's operating system via a `gethostbyname()` call. There is the opportunity for the operating system platform to also log this query. The platform normally does not operate a standalone DNS resolver, and often is configured by the local network provider with DNS resolvers to use. So my service provider is also privy to all my DNS activity. But it need not stop there. My service provider might farm out its queries to a recursive forwarder, so that it can avoid the overheads of running a full DNS resolver. Normally such forms of query indirection imply a loss of attribution, as such forwarded queries do not have any of my identifying details. Unless of course the resolver uses the EDNS0 Client Subnet option (RFC7871), in which case the forwarded queries still contain some critical details of my network.

All of these DNS queries can represent a lot of information even in these days of data intensity. Back in April 2015 Google reported that its public DNS servers deliver some 400 billion responses per day (https://webmasters.googleblog.com/2014/12/google-public-dns-and-location.html), and it appears that Google resolves some 12% of the total DNS load (http://stats.labs.apnic.net/dnssec), so that there were some 3 trillion DNS queries per day at that time. It can only be larger today.

THE RUMORS ARE TRUE. GOOGLE WILL BE SHUTTING DOWN PLUS—ALONG WITH HANGOUTS, PHOTOS, VOICE, DOCS, DRIVE, MAPS, GMAIL, CHROME, ANDROID, AND SEARCH—TO FOCUS ON OUR CORE PROJECT: THE 8.8.8.8 DNS SERVER.
Google

Figure 1 – XKCD: The Rumors are True (http://www.xkcd.com/1361/)

Not only is the DNS a chatty protocol that gratuitously sprays out information about user behaviours, it does so in an entirely open manner. DNS queries and their responses are unencrypted, and are sitting on port 53 in UDP and TCP. DNS queries can be easily intercepted, and, if DNSSEC is not being used, false answers can be inserted back and the client is none the wiser. In some countries DNS substitution appears to be relatively commonplace (http://www.potaroo.net/presentations/2013-08-29-facebook.pdf). Other countries have turned to DNS interception and blocking in response to problems associated with overloading IP addresses with virtual web hosting (http://www.potaroo.net/ispcol/2013-06/company.html).

DNS privacy has been a matter of some interest to the IETF, and changes are being proposed to the DNS protocol that would make it far harder to be used as a snooper's and censor's tool of choice. So what is going on to improve this situation and introduce aspects of privacy into the DNS.

QNAME Minimisation

In the DNS Operations Working Group, there has been work to perform what has been called query name minimisation in the DNS, resulting in a specification for "QNAME Minimisation" (RFC7816, March 2016). To quote from this document: "QNAME minimisation follows the principle [of] the less data you send out, the fewer privacy problems you have." In the example above the query to the root servers for the A record for `www.example.com` has two elements of gratuitous information: the fully qualified domain name and the query type. A more targeted query that does not gratuitously leak information is a query directed to the root name servers for the NS records for the `.com` domain. Similarly, the `.com` name servers would be queried simply for the name servers of `example.com` and so on (see Figure 2). In general, this approach is no less efficient than using a full query name at every point, and is equally capable of using cached information. The technique has exposed some inconsistencies with the handling of so-called empty non-terminal domain names, but the approach can be implemented in a robust manner, and it is a solid step in plugging a gratuitous information leak. It appears that the recently announced Knot DNS resolver from the CZNIC folk is one of the first DNS resolvers to implements QNAME minimisation (https://www.knot-resolver.cz) , as does Unbound (https://www.unbound.net) (from version 1.5.7, although I understand that the Unbound resolver implementation turns Qname Minimisation OFF by default).

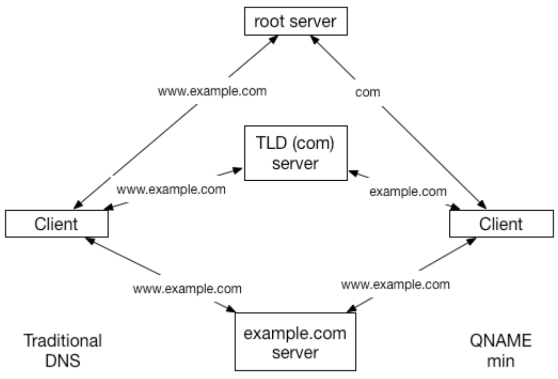

Figure 2 - The Intended Operation of Qname Minimisation

DNS and TLS

However, QNAME minimisation is only part of the privacy story. The open nature of DNS queries makes third party monitoring, interception and substitution incredibly easy, it appears. The DPRIVE Working Group of the IETF has been working on this topic, looking at ways for the DNS query and response interaction between a DNS client and a DNS resolver to be protected in some manner.

One issue here is whether to try and secure the current UDP-based resolution protocol, or head to a TCP-based approach where solutions already abound, typically based on Transport Layer Security (TLS). TLS conventionally requires a reliable transport channel, such as provided by TCP, and as such cannot be used directly to secure datagram traffic as used by UDP.

However, in some cases TCP is not seen as the optimal response to the problem. TCP attempts to ensure sequenced delivery, and in a message-oriented application, the loss of a message in TCP holds up the delivery of all subsequent messages until TCP can correct the data loss and deliver the lost message. This TCP "head of line blocking" can pose unacceptable overheads when using TCP to carry datagram-like message payloads. IPSEC could be seen as offering a cleaner fit when looking at securing a UDP-based application, but IPSEC is a kernel function rather than an application module, and its semantics apply at the IP later rather than as an attribute of the transport protocol. This makes it challenging to incorporate IPSEC into an application, and operate the cryptographic functions in user space.

One of the challenges has been to see if the functionality of TLS could be mapped into a datagram transport environment. Out of this consideration has come a new protocol, DTLS, which is an adaptation of the TLS function that can present to the application a datagram-like delivery function that does not require reliable transport services. DTLS can recover from packet loss and reordering, but it is intolerant of UDP packet fragmentation (RFC 6347). It is modelled upon TLS 1.2 and uses some explicit additional features that allow TLS to function over a datagram transport as distinct from a reliable stream transport. DTLS makes efforts to minimise the impact of the use of TLS on the DNS experience, particularly when compared to DNS-over-TLS-over-TCP. The major change is to require an initial DTLS handshake to set up a shared encryption state, and the use of cookies to re-use that state across multiple individual response/query interactions.

One of the main features of the current DNS protocol when used over UDP is how little shared state overhead each individual transaction incurs, resulting in a highly responsive can capable service. DNS over DTLS attempts, as far as possible, to preserve this simple query/response datagram exchange model but to do so in a manner where the client is using an encryption based on the validated credentials offered by the server. The current state of play of this specification is at https://tools.ietf.org/html/draft-ietf-dprive-dnsodtls-06. DTLS is intolerant of IP fragmentation, so the operation of DNS over DTLS is similar in design to the use of the Truncated bit in DNS over UDP as a signal to the client to repeat the query using TCP. Here the intended operation is that if a DNS over DTLS server has a response that is greater than the local Path MTU estimate, then the server should set the Truncated bit in its response, and this is to be interpreted by the client as a signal that the client should repeat the query using DNS-over-TLS.

The other option is to use conventional TLS, which is a TCP service. Much has been said on the use of TCP as a mainstream transport protocol for DNS, as distinct from its current intended role as a backup to UDP for large responses. It has been argued that the servers' TCP connection state overheads seriously impair their ability to handle large query loads, and the additional overhead of the protocol handshake would negatively impact on the user experience. On the other hand, it is argued that already the web is being used overwhelmingly as a short transaction service, and web servers appear to withstand the imposed load. It is also noted that the use of TCP is an effective measure against various forms of abuse that rely upon the ability to perform source address spoofing in UDP.

The Specification for DNS over Transport Layer Security (RFC7858, May 2016) is a relatively straightforward description, in that the transport service offered by TLS is effectively the same as that offered by TCP, but running the server's listener at TCP port 853, rather than port 443. There is perhaps one change here, and that is a suggestion for TLS session reuse: "In order to minimize latency, clients SHOULD pipeline multiple queries over a TLS session". For transactions between a client and a recursive resolver, the suggestion for session reuse makes some sense. For transactions between a client and authoritative name servers where the client is itself performing DNS resolution, this may not be so achievable. The choice of a distinguished TCP port is also interesting. If you wanted the secure channel DNS traffic to merge into all other traffic and pose a challenge to attempts to block this service, the temptation to use port 443 for DNS over TLS would be overwhelming (at least for me!). More information on the current state of clients and servers that support DNS over TLS can be found at https://portal.sinodun.com/wiki/display/TDNS.

Secure DNS over JSON

Last but not least there is the option to use an entirely different data encoding protocol, and here a recently announced service from Google is relevant. The server at https://dns.google.com performs a resolution function over TLS using port 443 with the results passed back as a JSON data structure. This can readily be transformed into an alternative form of gethostbyname() by the application substituting a web object retrieval for a conventional DNS query. This offers the caller some level of privacy from third party inspection and potential intrusion and censorship, although its unclear precisely what "privacy" means when you are sharing your DNS activity with Google!

Example script:

```
#! /usr/bin/env python
import json, requests

url = "https://dns.google.com/resolve"
params = dict(
    name='www.potaroo.net',
    type='A',
    dnssec='true'
)

resp = requests.get(url=url, params=params)
data = json.loads(resp.text)
print data[u'Answer'][0][u'data']
```

Concerns about data leakage is not limited to external forms of surveillance and interception. An appropriately paranoid application would not use the platform's DNS resolution service, as this would release the application's name queries into an uncontrolled environment where it may be logged accessed by the platform and other applications. In this case the application is not performing DNS resolution and validation itself, but in creating a secured channel to Google's resolution service across a TLS connection it can obtain some level of assurance that it is not performing a local leak of DNS information, and that with DNSSEC validation enabled the responses it receives have some level of assurance that they are genuine, assuming that the name being resolved is itself DNSSEC-signed.

Another similar approach is being constructed by the GetDNS project (https://getdnsapi.net). However, in this case it's not a secure channel to a recursive resolver that will resolve the application's queries, but a local validating resolver. This project currently supports DNS over TLS. This project operates as an open source project, and the GetDNS project page contains pointers to the code. A web application is built into the API, and a portal to a resolver implemented in this manner can be found at https://getdnsapi.net/query.html. There are some interesting trade-offs in this approach of pulling the DNS resolution function potentially all the way back into the application. The queries being made now have a source address of the local host, so the data that is leaked through the DNS queries can identify the local host. If an authoritative name server does not support a secure channel for queries using DNS over TLS, then the API will necessarily use an open unencrypted channel (at this stage the DNS over DTLS is not included in the GetDNS code base, but if someone wants to submit code …). On the other hand, the DNSSEC validation function can be performed locally as well by a GetDNS instance, so that the application is not forced to trust the authenticity of a bit flag in the response from a remote resolver. This way the application has direct control of the validation function, and direct knowledge of its outcome.

Between these two approaches there are further trade-offs that are apparent.

Making queries via a secure channel to a busy recursive resolver, and Google's Public DNS is about as busy as a DNS resolver can be, means that it is possible, to some extent, to hide behind the cache of such busy resolvers. As long as you are comfortable with sharing your DNS queries with Google, then to some extent you can use secured access to a DNS recursive resolver that intends to operate with integrity, accuracy and completeness. The secure channel is far harder to subvert and more resistant to efforts to eavesdrop upon the query stream.

If you are uncomfortable with this approach, then another option is to pull the name resolution function back into your platform and even back into the application itself using a framework such as GetDNS. The extent to which your queries may be readily visible to third parties, and the extent to which your query stream may be subverted in various ways is now dependent on the capabilities of the authoritative name servers. Without name server support for DNS over TLS and possibly DNS over DTLS, and without DNSSEC signed zones, the local DNS resolver may still be misled in ways that may not be readily detected. In this case the local resolver is powerless to fix this, as the privacy and protection mechanisms are now in the hands of the authoritative name servers and the zone admins that are queried by the local resolver.

What does all this mean?

While today that the open nature of DNS queries makes third party monitoring, interception and substitution incredibly easy, there are now some grounds to be optimistic and start to contemplate a DNS environment that preserves privacy and integrity.

By performing QName minimisation it is possible to radically reduce the level of leaked information coming from the DNS, and by wrapping up DNS queries and responses in a secured channel it is no longer trivial for third parties to monitor and intercept DNS queries and their responses on the wire.

If applications made use of services that would push local DNS query traffic into encrypted TLS sessions, such as the service being offered by Google, the result would be that much of today's visible DNS would disappear from view. Not only that, but it would make the existing practices of selective local inspection and intervention in the DNS resolution process far more challenging, if not infeasible. It may be even better if authoritative name servers were to also support queries over TLS and DTLS allowing a local host to take over the resolution function and still use encrypted query traffic services.

If this were to be coupled with widespread use of DNSSEC, then it's a somewhat different Internet from the one we have today. It's pretty obvious that national online censorship efforts will continue, and online monitoring and surveillance will also continue. But the ability to coopt the DNS into the role of an exceptionally cheap and simple means to achieve these ends will cease at some time if we collectively choose to head down this path for adding privacy and security the DNS.

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 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 an Internet researcher, as an ISP systems architect and a network operator at various times.

www.potaroo.net

Joao Luis Silva Damas is currently Senior Researcher at APNIC. Joao was also co-founder of Hivecast Inc, a DNS services company later sold to Dyn. Previously he worked at ISC (Internet Systems Consortium) as CTO overseeing technical developments. Earlier, he served as CTO at RIPE NCC and later founded Bond Internet Systems as consulting and research company. For around 7 years he organised the RIPE plenary program and launched the current RIPE program committee. In 2008, together with colleagues, launched ESNOG to bring together Spanish ISPs to interact with each other, an activity that continues to this day.

Disclaimer