December 2020
Geoff Huston

DNS Oblivion

Technical development often comes in short intense bursts, where a relatively stable technology becomes the subject of intense revision and evolution. The DNS is a classic example here. For many years this name resolution protocol just quietly toiled away. The protocol wasn't all that secure, and it wasn't totally reliable, but it worked well enough for the purposes we put it to. Even though its privacy was non-existent, and it leaked personal information like a sieve, we were largely unconcerned about the ramifications of this. Even when man-in-the middle attacks that performed DNS substitution became so commonplace that we started to rely on such DNS lies and withholding, we were still largely unconcerned. Obviously, it took a bit of a shock to wake us up out of our collective torpor over the DNS protocol, and the Snowden revelations provided such a jolt.

Since that time the IETF has taken some concerted action to make significant improvements in the privacy of a number of elements of the Internet operation, and the DNS protocol has been part of that overall effort. Work has progressed quickly on various approaches, particularly with the crucial stub-to-resolver link, where the end client's identity and the DNS name they are asking for are combined in the DNS query. The first set of responses were addressed by securing the channel used by the DNS, with DNS over TLS (DoT) and DNS over HTTPS (DoH) definitions that enclosed the DNS query and responses in a secure wrapper.

The problem with both DoH and DoT is that neither is all that satisfactory from a privacy standpoint. It is more of a compromise approach that poses a difficult question to me, as the end user. If I have to disclose both my identity and what DNS queries I'm making, then is it better to share all this information with my ISP through their DNS resolver, or should I share such critical information with Cloudflare or Google or any other of the open DNS resolver operators? If I can't hide the fact that I am the end client who is making this query, then who is least likely to compromise my privacy or abuse this privileged relationship? Let me restate this conundrum another way: If I have to compromise my privacy to a third party, the which third party represents the least risk to me now and in the future? It's a tough question, and the best answer not having to compromise your privacy at all.

ODNS

A group at Princeton University, Annie Edmundson, Paul Schmitt and Nick Feamster, together with Allison Mankin from Salesforce, has come up with an approach to break through this uncomfortable compromise with an approach they called "Oblivious DNS" (written up as https://datatracker.ietf.org/doc/html/draft-annee-dprive-oblivious-dns-00, July 2018, also a paper available at https://odns.cs.princeton.edu/pdf/pets.pdf).

The concept is delightfully simple, and it is intended to prevent the recursive resolver from knowing both the identity of the end point stub resolver and the queries that they are making. The stub takes the query name and encrypts it using a session key. Let's call this new query name $k\{qname\}$. The session key is encrypted using the public key of the target ODNS server, and appended to the encrypted query name. Let's call this $PK\{k\}$. The stub then appends the label of the oblivious DNS server domain (lets use **.ODNS** in this example). In the DNS the QNAME field consists of 4 sets of 63 bytes, which limits both the key size and encryption scheme used. For this reason, ODNS uses 16-byte AES session keys and

encrypts the session keys using the Elliptic Curve Integrated Encryption Scheme (ECIES). Once the session key is encrypted, the resulting value takes up 44 bytes of the QNAME field. The stub passes this query for ***k{qname}_PK{k}*.ODNS** as conventional DNS query to its recursive resolver. The recursive resolver is unaware of ODNS and treats the query as it would any other. The recursive resolver will then pass a query for this name to an authoritative server for **.ODNS**. To resolve this name the ODNS authoritative server will decrypt the session key (as it has the matching private key) and then use this to decrypt the query name. It can then use a conventional recursive resolution procedure to resolve the original query name. The response is encrypted using the provided session key. The ODNS server will then respond to the recursive resolver with the encrypted query name in the query section and the encrypted answer section that it has just generated. Upon receipt of this response, the recursive resolver will pass this to the stub resolver. The stub resolver uses its session key to decrypt the response. (Figure 1)

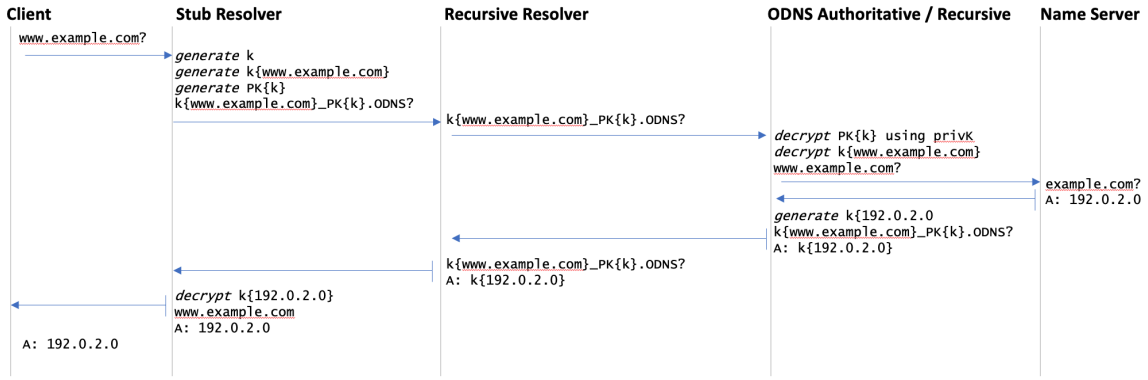

Figure 1 – Oblivious DNS

In the ODNS approach the ODNS stub resolver and the ODNS authoritative server collaborate to ensure that the recursive resolver is unaware of the actual query name. The recursive resolver is aware of the identity of the stub resolver, but not the query name. When the recursive resolver passes the query to an authoritative server for this 'special' domain, the authoritative server is aware of the query name (after decrypting it) but does not know the identity of the originating stub resolver, as the recursive resolver masks that information. When the authoritative server resolves this name, acting as a recursive resolver, other parts of the DNS may also be aware of the query name (although query name minimisation could mitigate this to some extent) but they are only aware of the identity of the authoritative server, and not the first hop recursive resolver nor the stub resolver.

Caching at the recursive resolver part of the ODNS authoritative server will still function in the same way as conventional DNS caching in recursive resolvers, however the recursive resolver that has been passed the encrypted query name should not cache the query and encrypted response.

ODNS is an intriguing approach to the problem, particularly in its use of existing recursive resolver infrastructure, but it's not the only approach out there at the moment.

ODoH

A different approach to the same problem, namely of trying to prevent any agent outside of the client's own domain from being able to match a DNS query to an endpoint identity, has been used by "Oblivious DoH" (https://tools.ietf.org/html/draft-pauly-dprive-oblivious-doh-02). It is slightly more involved than the original Oblivious DNS approach as it also uses an *Oblivious Proxy* in addition to an *Oblivious Target*, wrapping up the entire DNS transaction in an encrypted envelope. The *Oblivious Target* is, in DNS terms, just a recursive resolver, as there is no need to also act as an authoritative server because the steering of the query to the target is performed by DoH, rather than via the DNS itself.

A ODoH stub resolver uses DNS over HTTPS to pass queries to an *Oblivious Proxy*. The stub resolver takes the query, generates a session key (for the response), and encrypts both of these objects with the public key of the *Oblivious Target*. The HTTP wrapper includes the nomination of the target host and path in its query to the *Oblivious Proxy*. The proxy looks for the target host and path in the HTTP wrapper, and sends the encrypted payload to the target host, again using DoH transport (to the URI *https://targethost/targetpath*). The *Oblivious Target* can decrypt the original DNS message using its private key, and it will then act as a recursive resolver to resolve the query name in a conventional manner. The target then encrypts the DNS response using the client-provided symmetric session key and passes it back to the *Oblivious Proxy*, who then passes the encrypted object back to the client. The client can use the session key to decrypt the response. The original DNS query is left intact in this approach, and a two-level wrapper is used such that the outer wrapper, HTTS, is used in a hop-by-hop manner, and the inner wrapping of the DNS query and response shields the payload from all bar the intended recipient.

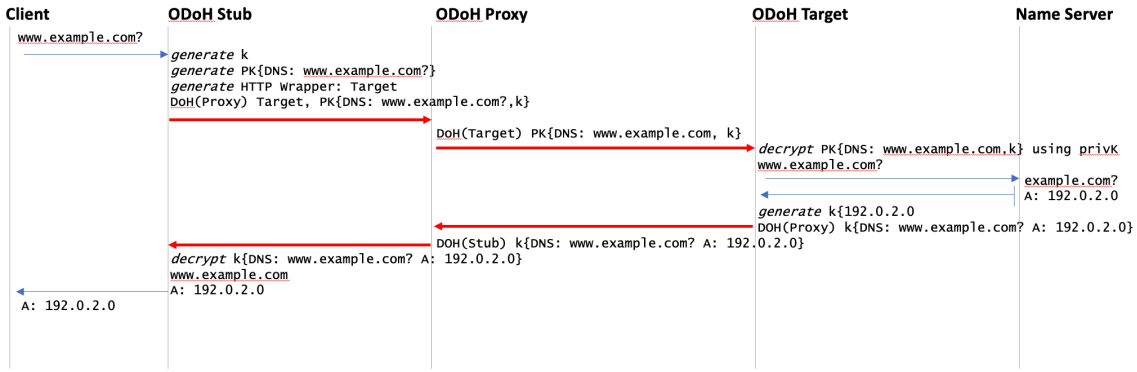

Figure 2 – Oblivious DoH

Comparisons

The ODNS approach is constrained in its approach as it is attempting to use the existing DNS recursive resolver infrastructure and therefore has a limited repertoire as to how it can encrypt the original query. The ODoH structure is slightly different. The disassociation of query and end client identity is made through deliberately breaking the stub-to-recursive DoH session into two sessions, stub-to-proxy and proxy-to-recursive. The proxy is only aware of the identity of the stub, but not aware of the query (as the entirety of the query object is encrypted using a key derived from the public key of the target resolver), while the target resolver is aware of the query but is unaware of the identity of the stub resolver as it only is aware of the identity of the proxy. It is absolutely essential that the proxy and the target cannot collude in this arrangement. It's probably best they be operated by quite distinct entities.

The advantage of the ODoH approach is that it is not attempting to shoehorn the encrypted information into a conventional DNS query packet format. The disadvantage is that it requires two dedicated agents, the *Oblivious Proxy* and the *Oblivious Target*, qualified by a strict proviso that the proxy and target must be operated by distinct entities who have no possibility of colluding. One could imagine the Proxy function being replaced by a TLS pass-through function. Don't forget that the DNS query itself does not contain a requestor identity field. In the DNS that information is provided by the outer layer, so that all DNS queries are by their very nature anonymous. The rationale for not using a TLS pass-through and the deliberate breaking of the query process into two steps is to permit the client an additional element of capability to vary the target and disperse its queries over a number of ODOH target servers if it so desires.

The advantage of the ODNS approach is that it only requires the provisioning of a target domain name and an associated authoritative server function, so a single party can be used for the service. The issue with the ODNS approach is that it is evident to any onlooker of the stub-to-recursive component that the client is making a ODNS query, and the query type is visible, even if the query name is opaque. This

could be readily addressed if the ODNS stub-to-recursive path used DoH rather than the conventional UDP/TCP transport for this hop.

Observations

It's not clear if either of these approaches will catch on, for some definition of what “catching on” means.

There is a strong aversion on the part of most users to playing with provided configurations for name resolution in their devices so both of these approaches might look like unlikely contenders for the mainstream of the DNS name resolution infrastructure.

However, you also need to remember that these days applications have an increasing interest in protecting the privacy of their users and are therefore interested in concealing their activity from other applications, from the host platform, from the local network, and any other third party at all! One can easily see as ODoH framework being used to support the name resolution requirements for an application where even the even the very existence of the application's name resolution operations has a requirement to be hidden. Doubtless such requirements exist out there and it may well be that this becomes a convention al part of an application's toolset in the near future, lifting the application out of the common name infrastructure and creating highly customised and well secured name frameworks for each application.

Is this application-level customisation of the DNS and name resolution operations what we meant when we talked about "DNS Fragmentation?" I guess so, and I suspect we are going to see more in terms of pre-provisioning and selective privacy measures that take DoH further along a path of what one could characterise as some degree of application level autonomy from the existing DNS infrastructure.

Disclaimer

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

Author

Geoff Huston AM is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net