Effectively Securing DNS

Will Archer, Julian Espada, Caleb Littlejohn, Deb Torres
warcher, jesp1999, calebl, dctorres

May 10, 2022

Abstract

The Domain Name System (DNS) is the conventional naming lookup scheme that maps alphanumeric URLs to IP addresses. DNS is inherently insecure in that it features no encryption or authentication, and yet this standard insecure variant is still used by the vast majority of modern systems. This paper discusses common variations of the DNS protocol, including DNSSEC, DoT, DoH, ODoH, and onion routing, and which configuration of these variations best meets the defined security goals. This includes preventing outside adversaries from easily being able to listen in on client requests and perform common malicious attacks. Also included are possible next steps to take to develop a more secure universal domain naming system.

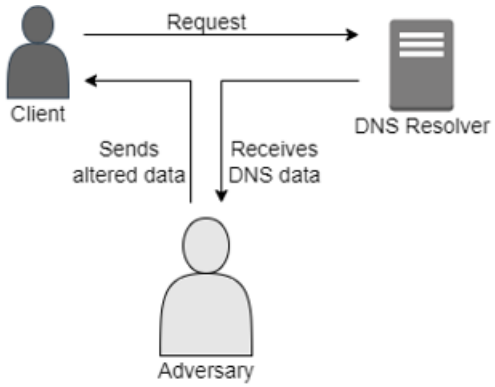
I. Introduction

The system of routing currently used for the internet is the Domain Name System (DNS). DNS is referred to as the phonebook of the internet, taking plaintext human readable names of websites and returning the IP address for a browser to be able to visit that site. This is important because the IP addresses for websites must be centralized, it would take an extraordinary amount of storage space for each device to keep its own directory of IP addresses and be difficult to distribute updates to maintain all of those directories. In order to make DNS responses efficient, the domain name space is structured as a hierarchy, meaning a resolver handling a DNS request first queries the root name server, then a Top Level Domain server (.com, .org, etc.), until the server with the IP address for the whole website requested is found and the IP address is returned by the resolver. A device can then cache this website and associated IP address to avoid future lookups in a short timeframe.

In describing DNS, there are a variety of similar terms used to name each portion of the system. In this work, the "resolver" takes the role of the nameserver, where a request in plaintext format is turned into a response of the corresponding IP address.

On the other side of DNS is a requester, user, or in this work a "client" who is looking to get the IP address of a plaintext website URL.

There are a variety of possible adversaries to a standard DNS name lookup. For example, one's internet service provider (ISP) might be utilizing a consumer's DNS payloads or metadata to decide to restrict traffic or extract information about the consumer. Given the ISP already has access to the content being served to the consumer, as well as the addresses from which this content is served, securing DNS against this party in particular might not be the most useful for analyzing the security of an alternate DNS protocol. Additionally, adversaries could be present in any part of the DNS transaction chain, whether in some proxy, nameserver, or the client itself, but an attack from an ISP cannot be protected against with changes at the protocol level. Since the purpose of this paper is to focus on solutions at the protocol level, these types of attacks are not considered in the motivating threat model. The adversary of focus is a "man in the middle" adversary who is able to read and alter the requests and responses sent between clients and DNS resolvers in order to perform malicious attacks.



Man in the Middle Attack

Without secure encryption of data, these adversaries are able to gather sensitive information on clients through a variety of methods. This eavesdropping third party is capable of silently listening to DNS traffic and can launch a number of attacks. They can choose to modify the contents of the packet returning from the resolver to the client. In this way, the adversary tricks, or "spoofs", the client into believing its contents are what the resolver intended to send. The adversary can use this method to direct the client to a malicious website or otherwise change the IP address accessed by the client. The adversary can also choose to simply listen in to the DNS traffic coming from a client to gather personal information about the client and their browsing history.

In analyzing defenses against these third party attacks, it is useful to define a security policy which an ideal, secure DNS implementation should be able to satisfy.

Namely, a secure DNS policy should be able to provide confidentiality and integrity to the client during their DNS transaction. Confidentiality is desired in that the client doesn't want their information to be leaked to parties not directly involved in the DNS transaction. This can be provided with sufficient encryption of traffic on the DNS line. To establish a measure of integrity and verify the data that the client receives from the resolver is actually what the resolver intended to provide, there needs to be a way to authenticate the data to verify responses have not been altered. Moving forward, this paper discusses how a combination of encrypting and authenticating traffic between a client and resolver is sufficient in mitigating attacks from the aforementioned threat model.

II. Existing Secure DNS Technology

There are currently many existing DNS variations that attempt to mitigate the ways that adversaries are able to attack. Some of the most popular are: DNSSEC, DNS over TLS (DoT), and DNS over HTTPS (DoH). There are additional variations on these as well, such as Oblivious DNS over HTTPS (ODoH) and onion routing which add additional steps to these protocols and provide a greater measure of client protection. What follows is discussion of some important aspects of each of these protocols and what key benefits they provide. None of these solve all of the problems with DNS, but instead focus on preventing specific problems from occurring, creating a more secure domain name system overall.

i. DNSSEC

DNSSEC stands for Domain Name System Security Extensions. The goal is to provide enhanced security to DNS requests and responses through authentication. In this protocol, messages containing DNS requests are signed by the client and DNS resolver in a way such that the adversary could not feasibly tamper with the information without being detected. This gives the client confidence that the information in response to a request comes from a valid resolver and has not been tampered with, satisfying the integrity goal, but it is important to note the protocol does not provide protection against an adversary viewing the requests, only ensuring they cannot be changed without the authentication check failing and the other party finding out. A downside to this protocol, however, is that this scheme adds significant complexity to both ends, with added steps to sign the request, sign the response, and check the signature is authentic. A primary feature of DNSSEC is that it is backwards compatible, meaning that if a queried nameserver for a particular DNSSEC request doesn't support DNSSEC, the system reverts to standard DNS and completes the

request without authentication, allowing for systems without this protocol to still operate at a basic level.

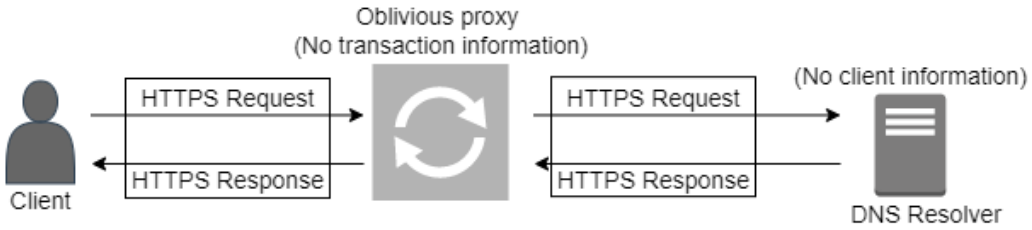
ii. DNS over TLS

DNS over TLS (DoT) is a network security protocol for encrypting DNS requests using the Transport Layer Security (TLS) protocol. Unlike DNSSEC, it provides protection against eavesdropping by encrypting the requests and responses, meaning the client can be confident an adversary is able to determine a negligible amount of information about the requests by just listening. However, the message contents are not protected from harm by an adversary who can encrypt their own message. This could lead to a "man in the middle" attack, where an adversary replaces the response from the resolver with a different response they have selected. Since the protocol does not authenticate the contents of the message like DNSSEC does, an altered or forged response would appear the same to the client, who does not have a way to distinguish this tampered message from an authentic response from the resolver. Also, another concern of DoT is that traffic via this protocol is very distinguishable from other types of web traffic, and it is restricted to only port 853, thus it is simple for an adversary to monitor that port and know they are seeing DNS packets. This can be observed in practice as some systems administrators are able to filter DoT traffic into a blacklist. Even though the packets are encrypted, such traffic analysis attacks are possible. Additionally, being in the transport layer requires more specifically designed software to handle these requests. For example, it cannot be used easily on top of an existing web browser which doesn't currently support these requests.

iii. DNS over HTTPS

DNS over HTTPS (DoH) encrypts DNS requests using Hypertext Transfer Protocol Secure (HTTPS). It provides many of the same security guarantees as DNS over TLS, in that the messages cannot be listened to by an eavesdropper. However, one key difference is that DoH traffic passes through the standard HTTPS port, 443. This means that DoH traffic will be mixed in with other HTTPS packets and an adversary would have greater difficulty executing a traffic analysis attack.

iv. Oblivious DNS over HTTPS



Oblivious DNS over HTTPS (ODoH) separates the knowledge of the identity of the client from the knowledge of the contents of the request itself. It does this by adding a proxy server to the DNS infrastructure. A client sends an encrypted DNS request to a proxy that passes the encrypted request on to the DNS resolver. Only the resolver can decrypt the request, formulate the response, encrypt the information and return the encrypted response to the proxy. The resolver will send the response to the proxy, which cannot decrypt the information, to match the response to the identity of the original client and return the encrypted response to the original client for decryption. The resolver never learns the IP address of the client, protecting the identity of the client from the resolver, and the proxy never learns what the request contained, which also protects the client from having their identity paired with the information they request. ODoH depends on confidence that the proxy and resolver servers will not collude. This is because these two bodies together carry the full information of the client and the body of the request. So long as these entities are separated and do not share information with each other, the client can be confident their identity is not paired with their requested information. The main vulnerability which arises in this instance is that an adversarial proxy can spoof messages which aren't actually coming from the resolver, such as responses to malicious DNS requests.

v. Onion Routing

Onion Routing is another method of securing the privacy and anonymity of the client from not only outside parties but also the resolver. As the name implies, there are many proxy layers to the Onion Routing system. These layers separate the client and the resolver, each representing a node in the path the request takes from one end to the other. The client encrypts the request multiple times, using the public keys of the route to the resolver in reverse order. For example, if the route is Client $\rightarrow$ ISP $\rightarrow$ A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ Resolver, the Client would first decide that route and encrypt the request with C's public key, then B's, then A's. At each node, the server "peels back" one layer by decrypting with the secret keys held by that node to find the next node in the path and sends the request there. Since each node only ever communicates

with the previous and next node in the predetermined route, it is very difficult for any single party to reconstruct useful information about the client or the request. This level of anonymity would be nice to have, but in practice is unfeasible. There is a large propagation delay and encryption cost to using Onion Routing that makes it unreasonable to deploy on a large scale when faster protocols with still reasonably good security are available.

III. Current System Development

Taking into account many of these modern secure DNS protocols, it is valuable to briefly discuss what major players in this space are currently doing. Two key players in DNS resolution are Google, providing 2.3% of worldwide DNS resolving, and Cloudflare, providing 14.26%. The systems that are currently feasible and in production today to serve as a reality check for what might be considered for new solutions.

Google is a strong proponent of the DoH scheme in much of its cloud technology, even going so far as to enable it by default in its Chrome web browser. Google's DNS resolver also currently supports DoH, DoT, and DNS over QUIC, with each having additional support for the DNSSEC structure. They have many more safeguards in place to prevent common attack patterns, such as rate-limiting queries to stop denial-of-service attacks, adding entropy to the requests so it is even more difficult for attackers to spoof clients, etc., but the primary security protocols remain DNSSEC and DoH.

Apart from Google, Cloudflare has released their own secure DNS with a focus on speed of resolution and security. They have also chosen to go the route of DNSSEC (when applicable) and DoH. With their massive server base and worldwide coverage, they are one of the fastest DNS resolvers, showing that DoH and DNSSEC is definitely a valid, scalable solution.

IV. Prevention Method

Out of the methods shown above, the best, currently feasible way to reach the defined security goals is to use the DNSSEC protocol carried via ODoH. The DNSSEC extension helps us achieve the goal of authentication of resolver response, which aligns with preservation of integrity of name resolution. Carrying this DNSSEC payload over HTTPS gives us widely-supported encryption of data between resolver and client, which aligns with the goal of communication confidentiality. The choice of HTTPS rather than TLS is largely subjective, as they are of comparable efficiency, but DoH traffic is harder to distinguish in aggregate from other HTTPS traffic,

leading to greater protection against traffic filtration attacks. In addition, the current availability of web browsers that support HTTPS over TLS would make a transition to this new DNS much easier. The choice of having DNS traffic communicated "obliviously" through a trusted proxy takes away the ability for the resolver to collect data about the client, which is an additional optional layer of security on top of the aforementioned layers.

One important thing to note is the network propagation delay that would be introduced with such a proxy is non-negligible, making this a non trivial choice, and generally should depend on the use-case. Onion Routing is not included in the ideal solution because, while it removes the need for a single trusted proxy, it adds a considerable amount of delay to the request, far more than would be deemed acceptable for most modern applications of DNS. This raises concerns that a protocol implemented using DNSSEC over ODoH would be too slow on a wide scale, but the data collected by sending requests through Google's DoT, DoH, DoT with DNSSEC, and DoH with DNSSEC resolvers show that once any sort of authentication/encryption is added, the latency appears to be about the same with any variation of the above (see Appendix).

V. Unsolved Problems

The proposed system is able to satisfy the defined security goals with respect to the threat model, but additional steps need to be taken to provide a secure scheme against more generalized adversaries and improve faults in the existing schemes. For starters, existing secure DNS schemes are non-negligibly slow, especially when compared to the standard insecure DNS scheme. As depicted in the data in the appendix, while each of the secure protocols performs quickly with respect to each other, the resolution time of using many repetitions of these schemes is very sluggish in comparison to non-encrypted DNS. This problem could largely be solved by improving the speed of encryption and authentication algorithms or hardware more generally.

Additionally, the more generalized adversary could be a party actively involved in the DNS resolution, such as a proxy or DNS nameserver. For instance, the client might be concerned about the DNS nameserver, a suspicious ODoH proxy, or their ISP learning too much about their browsing traffic through DNS analysis. The first two of these are currently best solved via onion routing, where the identity of the requester is obfuscated by the multitude of parties on the line, but this routing scheme is incredibly latency-inducing, to the point where websites loading external assets have drastically increased load times, occasionally multiple minutes long.

One of the best solutions to more generalized DNS insecurity is simply for more clients across the world to adopt existing secure DNS implementations and deprecate their use of classic DNS. This would prevent the majority of currently occuring DNS-

based attacks, as the attack surface for DNS traffic would decrease drastically. This process could be further expedited by adoption of more secure DNS standards and requirements by major players in the industry, such as the IETF, government agencies, or large tech corporations. Also, many users of DNS worldwide are simply not concerned about this facet of their online security or don't have the software/hardware means to support these protocols commonly. However, even though there will be issues with any sort of change to DNS, other technologies have undergone substantial security improvements in recent decades and a gradual shift towards more secure DNS implementations can be expected over time.

VI. Conclusion

A secure DNS protocol is necessary to achieve the confidentiality and integrity goals laid out in this report. Of the protocols examined, the best combination to increase security is the DNSSEC protocol carried via ODoH. The DNSSEC extension to authenticate the resolver response creates integrity. Carrying this DNSSEC payload over HTTPS encrypts the data between resolver and client, which adds communication confidentiality. The collected data shows the existing secure DNS protocols that include encryption do add some delay as compared to regular DNS, which is going to also be the case for this solution. This calls for faster encryption if the efficiency of the protocol is important. Ultimately, the limit of this proposed solution will be in garnering widespread adoption. The existing protocols are not more popular because the barriers to change to these new systems are too high for many people, in that DNS isn't "broken" and the average user doesn't have enough awareness to want increased security.

VII. Acknowledgments

The authors would like to thank the course staff for 6.857: Applied Cryptography and Security, particularly Kyle Hogan for introducing us to many of the existing security concerns and solutions in modern DNS technology. Their feedback and assistance during the process of writing this paper has been incredibly beneficial to this work.

A statement requiring citation [**?**].

References

[1] "Usage Statistics and Market Share of Google as DNS Server Provider, May 2022." n.d. W3Techs. Accessed May 3, 2022. https://w3techs.com/technologies/details/dn-google.

[2] “Cloudflare DNS - Market Share, Competitor Insights in Domain Name Services.” 2022. Slintel. https://www.slintel.com/tech/domain-name-services/cloudflare-dns-market-share.

[3] “Security Benefits | Public DNS.” n.d. Google Developers. Accessed May 3, 2022. https://developers.google.com/speed/public-dns/docs/security.

[4] Baheux, Kenji. 2020. “A safer and more private browsing experience with Secure DNS.” Chromium Blog. https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html.

[5] “Overview · Cloudflare 1.1.1.1 docs.” n.d. Cloudflare Developers. Accessed May 3, 2022. https://developers.cloudflare.com/1.1.1.1/.

[6] Singanamalla,. n.d. “Oblivious DNS over HTTPS (ODoH): A Practical Privacy Enhancement to DNS.” https://petsymposium.org/2021/files/papers/issue4/popets-2021-0085.pdf.

[7] Eric Kinnear et al., “Oblivious DNS Over HTTPS,” Oblivious DNS over HTTPS, January 26, 2021, https://www.ietf.org/archive/id/draft-pauly-dprive-oblivious-doh-04.htmlname-denial-of-service.

A. Appendix

Additional data gathering on DNS protocol response times and the standard deviations. Data was gathered by sending 10,000 DNS packets to Google's DNS server "8.8.8.8" to resolve the domain name "example.com" by means of DoH, DoT, standard DNS, and DoH/DoT but with the DNSSEC extension.

Trial 1

	Average Response Time (ms)	Standard Deviation (ms)
TLS	56.3628	25.6625
HTTPS	58.4531	43.6243
TLS with DNSSEC	56.6060	18.7599
HTTPS with DNSSEC	57.5115	23.9572
DNS	11.6313	53.2297

Trial 2

	Average Response Time (ms)	Standard Deviation (ms)
TLS	56.5210	22.8113
HTTPS	56.8634	21.5837
TLS with DNSSEC	56.3115	21.0592
HTTPS with DNSSEC	56.5826	17.5872
DNS	10.4235	13.8591

Trial 3

	Average Response Time (ms)	Standard Deviation (ms)
TLS	55.8628	22.0432
HTTPS	55.4707	12.3757
TLS with DNSSEC	55.5600	20.9224
HTTPS with DNSSEC	56.0353	16.5000
DNS	9.7803	10.3828

Trial 4

	Average Response Time (ms)	Standard Deviation (ms)
TLS	54.9513	14.3548
HTTPS	55.2579	10.2316
TLS with DNSSEC	54.5411	9.8065
HTTPS with DNSSEC	55.2481	10.6646
DNS	9.6678	10.8149

Trial 5

	Average Response Time (ms)	Standard Deviation (ms)
TLS	54.4089	7.4581
HTTPS	54.9195	4.6044
TLS with DNSSEC	54.5059	7.4432
HTTPS with DNSSEC	55.3030	6.9339
DNS	9.8177	6.9934