Oblivious DNS: Practical Privacy for DNS Queries

Published in PoPETS 2019

Paul Schmitt
pschmitt@cs.princeton.edu
Princeton University

Anne Edmundson
annie.edmundson@gmail.com
Princeton University

Allison Mankin
allison.mankin@gmail.com
Salesforce

Nick Feamster
feamster@cs.princeton.edu
Princeton University

ABSTRACT

Virtually every Internet communication typically involves a Domain Name System (DNS) lookup for the destination server that the client wants to communicate with. Operators of DNS recursive resolvers—the machines that receive a client's query for a domain name and resolve it to a corresponding IP address—can learn significant information about client activity. Recognizing the privacy vulnerabilities associated with DNS queries, various third parties have created alternate DNS services that obscure a user's DNS queries from his or her Internet service provider. Yet, these systems merely transfer trust to a different third party. We argue that no single party ought to be able to associate DNS queries with a client IP address that issues those queries. To this end, we present Oblivious DNS (ODNS), which introduces an additional layer of obfuscation between clients and their queries. To do so, ODNS uses its own authoritative namespace; the authoritative servers for the ODNS namespace act as recursive resolvers for the DNS queries that they receive, but they never see the IP addresses for the clients that initiated these queries. Our experiments using a prototype show that ODNS introduces minimal performance overhead, both for individual queries and for web page loads. Critically, we design ODNS to be compatible with existing DNS infrastructure.

1 INTRODUCTION

Almost all communication on the Internet today starts with a Domain Name System (DNS) lookup. Before communicating with any Internet destination, a user application typically first issues a Domain Name System (DNS) lookup, which takes a domain name (*e.g.*, `google.com`) and returns an IP address for the server that the client should contact. Today, the DNS requires the user to place tremendous trust in DNS operators, who can see all of the DNS queries that a user issues. Whether the operator is an Internet service provider (ISP) or a third party is less concerning than the fact that some single operator can observe and retain this sensitive information. This paper presents a system, called ODNS, which attempts to solve this problem.

As DNS operates today, queries and responses are viewable as plaintext at the recursive resolver, even if the client is using an encrypted channel between it and the recursive resolver. As a result, they can reveal significant information about the Internet destinations that a user or device is communicating with. For example, the domain names themselves reveal the websites that a user visits. Previous work has also demonstrated that DNS lookups can identify the websites that a user is visiting even when they are using an anonymizing service such as Tor [1]. Recursive DNS resolver operators can readily associate and track client identities (*i.e.*, IP addresses) along with information about their DNS queries, creating a fundamental point of privacy risk.

A user's ISP often operates the user's default recursive DNS resolver, giving the ISP potentially extraordinary access to DNS query information. To mitigate this risk, several entities, including Google, Cloudflare [7], and Quad9 [4] operate "open" recursive DNS resolver services that anyone can use as an alternative to their ISP's DNS recursive resolver. Yet, when a user switches to such an alternate resolver, the privacy problem isn't solved; rather, the user must then trust the operator of the open recursive rather than their ISP. Essentially, the user must decide whether they trust their ISP or some other organization—some of which are even in the business of collecting data about users.

Other approaches have layered encryption on top of DNS [2, 3, 5]. This work takes a different tack. Instead of merely shifting the trust anchor from an ISP to some other third party, we seek to prevent a recursive resolver from associating client identities with the queries they make. To do so, we design, implement, and deploy Oblivious DNS (ODNS), which (1) obfuscates the queries that a recursive resolver sees from the clients that issue DNS queries; and (2) obfuscates the client's IP address from upper levels of the DNS hierarchy (*i.e.*, the authoritative servers). ODNS operates in the context of the existing DNS protocol, allowing the existing deployed infrastructure to remain unchanged. ODNS decouples client identity from queries by leveraging the behavior of the global DNS system itself. A client sends an encrypted query to a

recursive resolver, which then forwards the query to a ODNS resolver (an authoritative DNS server that can resolve ODNS queries). The recursive resolver never sees the domains that the client queries, and the ODNS resolver never sees the IP address of the client.

We design ODNS with performance being paramount. Tunneling DNS over the Tor network provides users with anonymity, but Tor's fundamental design introduces substantial network latency to the end-to-end path. As DNS underpins almost all web traffic, increased DNS lookup latency would present an outsized impact of web performance [6].

ODNS provides benefits for both the recursive resolver operators as well as users of the system. ODNS reduces the information that operators are able to know as they cannot associate queried domains with client identity, making them less-valuable targets. In essence, the operators are oblivious to their client's requests. Likewise, users of ODNS benefit in that they are no longer required to trust that their recursive resolver has their best interests at heart. Instead, users can be assured that all DNS infrastructure beyond the stub is unable to link their identity with their querying behavior.

We design ODNS to be immediately deployable alongside existing DNS infrastructure. We implement this functionality in a prototype ODNS stub and ODNS resolver in Go. ODNS's privacy enhancements come at a performance cost. Our trace-driven evaluation shows that any individual uncached DNS lookup is slower due to two factors: 1) our cryptographic operations add roughly two milliseconds to the lookup time; and 2) the round-trip time between the client and the ODNS resolver is added for each lookup. We mitigate the impact of round-trip time latency by designing ODNS to use anycast. Ultimately, our evaluation shows that the performance overhead on web page load times is negligible. Additionally, we reduce the traffic burden placed on existing recursive resolvers by implementing a cache at the stub resolver.

2 OBLIVIOUS DNS (ODNS)

Figure 1 summarizes the ODNS design. ODNS operates similarly to conventional DNS, but alters two components: (1) each client runs a modified stub resolver; and (2) an organization operating ODNS runs an authoritative name server (ODNS resolver) that also acts as a recursive DNS resolver.

The recursive DNS resolver has access to the client IP address, but it never sees the domains that it queries. ODNS requires the client to use a custom local stub resolver, which hides the requested domain from the recursive resolver. The ODNS stub resolver encrypts the original DNS query and the key used for encryption before it appends a plaintext ODNS-specific domain (*e.g.*, `.odns`) to the query, which causes the recursive resolver to forward the encrypted domain name on to the appropriate authoritative server (an ODNS resolver).

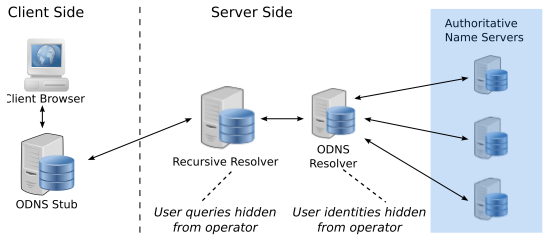

Figure 1: Overview of components in ODNS.

The recursive DNS resolver receives the request from the client stub, but cannot identify the genuine domain. It queries the appropriate TLD nameserver and forwards the request to the ODNS resolver. The steps involved in answering a client's DNS request proceed as follows:

(1) When a client issues a DNS request, the local stub resolver generates a symmetric session key, encrypts the domain name with the session key, encrypts the session key with the ODNS resolver's public key, and appends the ODNS domain (*e.g.*, `.odns`) to the encrypted query. ($\{\text{www.foo.com}\}_k$.odns.) The stub also appends the session key encrypted under the ODNS resolver's public key ($\{k\}_{PK}$)
(2) The stub sends the query to the recursive resolver, which then sends it to the authoritative nameserver for the specified ODNS domain.
(3) The ODNS resolver decrypts the session key, which it then uses to decrypt the genuine domain in the query.
(4) The ODNS resolver forwards a recursive DNS request to the appropriate name server for the plaintext domain, which then returns the answer to the ODNS resolver.
(5) The ODNS resolver returns an encrypted answer to the client's recursive resolver.

3 CONCLUSION

DNS queries can reveal personal information and allow DNS resolvers to associate such information with client IP addresses. Users are required to place trust in their recursive DNS resolver. Privacy-focused, third party DNS resolvers simply shift the trust without alleviating the fundamental information exposure. In this work, we present ODNS, a system that decouples client IP address from DNS queries, removing the need for trust altogether as no DNS infrastructure outside of the user network is able to obtain both pieces of information. ODNS is designed to be fully-compatible with existing DNS infrastructure and requires only minimal changes. Our evaluation of ODNS reveals that latency overhead is minimal, performance for user web traffic is acceptable, and minimal impact on recursive resolver traffic.

REFERENCES

[1] Benjamin Greschbach, Tobias Pulls, Laura M. Roberts, Philipp Winter, and Nick Feamster. 2016. The Effect of DNS on Tor's Anonymity. *CoRR* abs/1609.08187 (2016). arXiv:1609.08187 http://arxiv.org/abs/1609.08187

[2] Paul Hoffman and Patrick McManus. 2018. *DNS Queries over HTTPS (DOH)*. Internet-Draft. http://www.ietf.org/internet-drafts/draft-ietf-doh-dns-over-https-08.txt http://www.ietf.org/internet-drafts/draft-ietf-doh-dns-over-https-08.txt.

[3] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. 2016. *Specification for DNS over Transport Layer Security (TLS)*. RFC 7858.

[4] Quad9 2017. Quad9. https://quad9.net/. (2017).

[5] T. Reddy, D. Wing, and P. Patil. 2017. *DNS over Datagram Transport Layer Security (DTLS)*. RFC 8094.

[6] Srikanth Sundaresan, Nick Feamster, Renata Teixeira, and Nazanin Magharei. 2013. Measuring and Mitigating Web Performance Bottlenecks in Broadband Access Networks. In *Proceedings of the 2013 Conference on Internet Measurement Conference (IMC '13)*. Barcelona, Spain.

[7] What 2018. What is 1.1.1.1? https://www.cloudflare.com/learning/dns/what-is-1.1.1.1/. (2018).

3