Boost DNS Privacy, Reliability, and Efficiency with opDNS Safe Query Elimination

Marcel Waldvogel Thomas Zink
University of Konstanz, Konstanz, Germany
<first>.<last>@uni-konstanz.de

***Abstract*—SRV records, DNSSEC, and DANE among others fortify the Domain Name System as the central information hub behind the Internet. Largely hidden from the end user, an increasing number of protocol and trust decisions are contingent on DNS. Neglect or attacks on DNS have much more impact today than ever, now endangering security far beyond denial of service. Opportunistic Persistent DNS (opDNS) addresses these problems by abandoning pessimistic caching and eliminating unnecessary traffic. Today's DNS infrastructure relies on the hosts forgetting and refreshing DNS records in relatively short time. In conjunction with TLS, opDNS greatly reduces the number of queries and in turn increases privacy, reliability, and efficiency. Even with DNS lookups all but eliminated for frequently visited secure services, changes to the server addresses will be recognized almost immediately, unlike standard DNS. We will show how end systems can take advantage of opDNS without having to wait for support by server operators or application developers, enabling the most effective way of deployment.**

I. Introduction

When connecting to networks anywhere in the world, your end system sends its DNS queries through a server provided by the local operator. Neglect, misconfiguration, and active spying or modification of DNS records becomes disastrous, yet is trivial and extremely resource-efficient. One of the first things your device will do when connected to the new network is re-establishing connections to mail servers, web applications, instant messaging servers, and many more push services.

DNS security is a major problem, as DNS is so valuable, ubiquitous and important, but also vulnerable:

1) The network provider immediately recognizes many users due to the highly unique signature created by the set of queries. Queries also provide most revealing information about professional and personal affiliations, as well as personal interests.
2) DNS misconfiguration, attacks, or bugs will cause your applications and clients to fail or connect to the wrong site [7]. DNSSEC could mitigate these effects, unfortunately the vast majority of end systems do not currently validate it [3].
3) Not only black-hats cause blackouts. Nation states all around the globe have used DNS blocking to restrict access to certain services and perform censoring. The structure of DNS allows easy censorship and outages.
4) The number of requests, especially on slow or error-prone links, can significantly slow down operations, time better spent on the actual connection to the services.

The response to these and other threats [5], [6] has been to authenticate the data using DNSSEC [2] and work toward encrypted channels through VPNs, Tor, DNSCurve [1], or Confidential DNS [9]. DNS encryption, however, is unlikely to become an immediate solution for all problems.

The best way to avoid exposure of DNS queries and be immune to wrong answers is not to perform any DNS queries. This is the goal of our approach, *Opportunistic Persistent DNS* (opDNS). Orthogonal to DNS encryption and building on top of existing TLS and optionally DNSSEC, it significantly reduces the common and most identifying – as well as most vulnerable – DNS queries without sacrificing accuracy.

By using a slightly modified local cache and name resolver, and collaborating with an enhanced TLS library, unmodified applications can talk to unmodified servers with greatly reduced privacy exposure and risk of being subject to malicious or accidental DNS problems. DNS records are optimistically kept by the local cache for sites whose identity can be verified using TLS and which have been proven to be stable. Applications and servers aware of opDNS can take advantage of additional mechanisms, such as pushing authenticated DNS updates. In addition to increase privacy and reduce impact of outages, opDNS also greatly reduces the load on the DNS infrastructure. We strongly believe that increased privacy coupled with additional functionality are strong incentives for users and network operators alike.

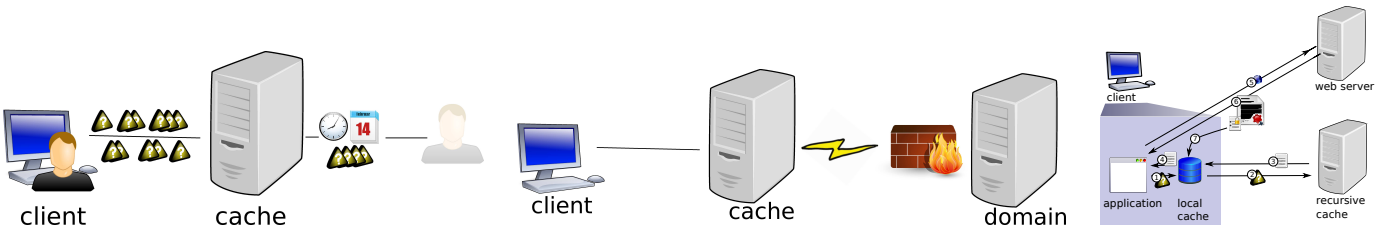
II. DNS Insights

The collection of a user's DNS queries provides a discriminating database of his interests, services used, and browsing behavior and allows user identification (Figure 1(a))[4]. Simply blocking name servers or certain DNS records makes name resolution impossible and prevents users from navigating to their target sites (Figure 1(b)). This is due to the lack of cache update and misuse for load balancing. Server operators prefer frequent DNS lookups instead of keeping possibly outdated information in cache, which would render a service essentially unreachable. In contrast, prominent servers tend to change their addresses only every few years, if at all. Nevertheless, an address change could become unavoidable without notice in a few minutes or days.

The simplest method to make traffic less revealing is to not generate the traffic at all. This applies even more to DNS, where low cache times (TTL) require the clients to send request again every few minutes to few hours.

III. Basic opDNS operation

An overview of the operation of opDNS is shown in Figure 1(c). A local cache on the user's machine retains a copy of recently queried DNS records. During the lifetime of



(a) Collecting DNS queries with date and time stamp allows tracking of users, their browsing behavior and can lead to user identification.

(b) Blocking authoritative name servers or eliminating specific records prevents successful lookup and ultimately leads to no connection to the target host.

(c) opDNS uses a local cache to reduce DNS traffic. Records are updated securely by the web servers using TLS.

Fig. 1. DNS problems and the opDNS approach.

the record, as indicated by the TTL value, the local cache is used normally.

After the TTL expires, the behavior deviates from the current standards: Records related to TLS services will be kept further, but with a special 'expired' flag set. When a TLS-enabled application asks for the record, the expired and flagged entry will be returned. The application will then proceed to connect to the server with caution, using shorter timeouts and strong certificate test. If this fails, the expired entry is flushed and the procedure is repeated.

IV. Transparent opDNS

The above basic process requires modifications to the application. This will unacceptably slow down deployment. A connection setup for a TLS-enabled client application will do the following steps, among other things:

1) Using the server name, obtain the address of the server using gethostbyname(), getaddrinfo(), or a call to the DNS resolver library
2) Set up a socket connection using connect()
3) Start the TLS handshake (e.g. SSL_connect())
4) Verify that the certificate actually belongs to the named server from step 1 (e.g. X509_check_host())

By hooking into the system libraries [8], the application's privacy awareness can be increased fully transparently. The opDNS library intercepts several of the above library functions and their relatives. By observing the calls to e.g. getaddrinfo() and X509_check_host(), it can determine that the application uses TLS to verify the server.[1]

An application that has proven to contact this host only using TLS, will in the future also be served expired DNS records. A connection error or certificate verification failure will wipe the expired entry from the cache.

V. Demonstration

As part of the demonstration, we will use devices which have the current 'pessimistic' DNS caching changed into the optimistic variant, opDNS, to

1) enhance privacy by disclosing as little personal information as possible,
2) reduce the effect of outages by localizing name resolving, and
3) improve client and server performance by reducing overall DNS traffic.

We will show that a device which has previously connected to a service in a *good* environment will be able to continue working even in *adverse* DNS environments with logging, blocking, or modification. We will also show that a device operating in a *good* environment will react quicker to changes in DNS, flushing its cache before the DNS record's TTL has expired, if necessary, while still eliminating most of the normal DNS lookups. All the tests will be run with off-the-shelf applications and servers; the only addition being a small linux library, which can be enabled or disabled as necessary. The library will be published under a free and Open Source license to enable instant deployment. Participants are welcome to connect their devices to our environments and experience the effects firsthand.

[1] Of course, failures between these calls, such as the server being unreachable for connect(), should be taken into account.

References

[1] "DNSCurve: Usable security for DNS," accessed 2014-04-14. [Online]. Available: http://dnscurve.org

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS security introduction and requirements," RFC 4033 (Proposed Standard), Internet Engineering Task Force, Mar. 2005. [Online]. Available: http://tools.ietf.org/html/rfc4033

[3] G. Huston, "DNS, DNSSEC, and Google's public DNS service," http://www.circleid.com/posts/20130717_dns_dnssec_and_googles_public_dns_service/, 7 2013, accessed 2014-10-31.

[4] S. Krishnan and F. Monrose, "DNS prefetching and its privacy implications: When good things go bad," in *Proceedings of USENIX LEET '10*, 2010.

[5] S. Ragan and S. Writer, "Three types of DNS attacks and how to deal with them," http://www.csoonline.com/article/2133916/malware-cybercrime/three-types-of-dns-attacks-and-how-to-deal-with-them.html, accessed 2014-04-14.

[6] P. Rubens, "How to Prevent DNS Attacks," http://www.esecurityplanet.com/network-security/how-to-prevent-dns-attacks.html, accessed 2014-04-14.

[7] H. Specht, "Major internet outage in china," Jan. 2014. [Online]. Available: http://apmblog.compuware.com/2014/01/22/major-internet-outage-in-china/

[8] M. Waldvogel, "TLS Interposer," https://netfuture.ch/tools/tls-interposer/, accessed 2014-04-14.

[9] W. Wijngaards and G. Wiley, "Confidential DNS," Internet-Draft (work in progress), Internet Engineering Task Force, Sep. 2014. [Online]. Available: http://tools.ietf.org/html/draft-wijngaards-dnsop-confidentialdns-02