T-DNS: Connection-Oriented DNS to Improve Privacy and Security (poster abstract)

Liang Zhu[1] Zi Hu[1] John Heidemann[1]
Duane Wessels[2] Allison Mankin[2] Nikita Somaiya[1]
1: University of Southern California 2: Verisign Labs

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Applications*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Public networks*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*

Keywords: Domain Name System (DNS), Transport Layer Security (TLS), privacy, security, network protocols, performance

1. INTRODUCTION

DNS is the canonical protocol for connectionless UDP. Yet DNS today is challenged by eavesdropping that compromises privacy, source-address spoofing that results in denial-of-service (DoS) attacks on the server and third parties, injection attacks that exploit fragmentation, and size limitations that constrain policy and operational choices. We propose *T-DNS* to address these problems. It uses TCP to smoothly support large payloads and to mitigate spoofing and amplification for DoS. T-DNS uses transport-layer security (TLS) to provide privacy from users to their DNS resolvers and optionally to authoritative servers. Expectations about DNS suggest connections will balloon client latency and overwhelm servers with state, but our evaluation shows costs are modest: end-to-end latency from *TLS to the recursive resolver is only about 9% slower* with UDP to the authoritative server, and 22% slower with TCP to the authoritative. With diverse traces we show that frequent connection reuse is possible (60–95% for stub and recursive resolvers, although half that for authoritative servers), and after connection establishment, we show TCP and TLS latency is equivalent to UDP. With conservative timeouts (20 s at authoritative servers and 60 s elsewhere) and conservative estimates of connection state memory requirements, we show that *server memory requirements match current hardware*: a large recursive resolver may have 24k active connections requiring about 3.6 GB additional RAM. We identify the key design and implementation decisions needed to minimize overhead: query pipelining, out-of-order responses, TLS connection resumption, and plausible timeouts.

Here we summarize our approach and poster, and expand on evaluation, alternatives, and deployment elsewhere [17];

2. PROBLEM STATEMENT

Our goal is to reduce the limitations that result from DNS' current optimization around a single-packet exchange.

Need for DNS Privacy: Traditionally, privacy of Internet traffic has not been seen as critical. However, recent trends in DNS use, deployment and documentation of widespread eavesdropping increase the need for query privacy [3]. First, end-user queries are increasingly exposed to possible eavesdropping, through use of third-party DNS services such as OpenDNS and Google Public DNS, and through open networks such as WiFi hotspots. Second, the presence of widespread eavesdropping and misdirection is now well documented, for government espionage [7], censorship [1], and criminal gain [12]. Finally, ISPs have recognized the opportunity to monetize DNS typos, redirecting non-existing domain responses (NXDOMAIN hijacking), a practice widespread since 2009 (for example [13]).

Need for Sender Validation: *Uncertainty about the source address of senders* is a problem affecting DNS servers and others on the Internet. Today source IP addresses are easy to spoof, allowing botnets to mount denial-of-service (DoS) attacks on DNS servers directly [10], and to leverage DNS servers in amplification attacks [16] against third parties. Specific work-arounds to DNS' role in DoS attacks exist, and rate-limiting helps. T-DNS would greatly reduce the vulnerability of DNS to DoS in a general way. Well established techniques protect DNS servers from TCP-based DoS attacks [6, 15], and TCP's connection establishment precludes source address spoofing, eliminating amplification attacks, and webservers developed solutions for TCP-specific risks [17].

Avoiding Arbitrary Limits to Response Size: *Limitation in payload size* is an increasing problem as DNS evolves to improve security. Without EDNS, UDP DNS messages are limited to 512 B. With EDNS, clients and servers may increase this limit (4096 B is typical), although this can lead to IP fragmentation raising its own problems [11]. UDP packet limitations expose DNS to fragmentation attacks and security problems [8]. Of greater concern, UDP size constraints have a corrosive effect on policy decisions. Three examples include limits to the number of root servers, limits to sizes of DNSSEC keys and raising concerns about key rollover. By replacing current 1024-bit RSA signatures with longer ones in a trace captured at one server for `.com`, over the 13.5M DNSSEC enabled responses, we find that with a 2048-bit Zone Signing Key (ZSK), 5% of DNSSEC responses, almost *all* NXDomain responses, and some DNSKEYs during rollover will suffer IP fragmentation.

3. DESIGN AND IMPLEMENTATION

T-DNS uses in-band TLS negotiation (our addition). We identify implementation choices needed for good performance.

Upwards TLS Negotiation: IETF encourages in-band upgrade to TLS (not new ports); this approach is the current preference for many protocols, including IMAP and SMTP. We therefore propose [9] a new EDNS0 extension [5] to ne-

SIGCOMM'14, August 17–22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2836-4/14/08.
http://dx.doi.org/10.1145/2619239.2631442.

gotiate the use of TLS. We suggest a new "TLS OK" (TO) bit as an EDNS0 option. A client requests TLS by setting this bit and sending a DNS query. A server that supports TLS responds with the same bit set, then both client and server transition to a TLS handshake. The DNS query made to start TLS negotiation is sent without encryption so it should not disclose information. We recommend a distinguished query for name "STARTTLS," type TXT, class CH, analogous to current support queries.

Implementation choices are critical to good performance. *Pipelining* is the ability to send several queries before the responses arrive. It avoids round-trip delays by the stop-and-wait alternative. Batches of queries are common: recursive resolvers with many clients will have concurrent queries to popular authoritative servers like `.com`, and 62% of web pages have 4 or more unique domain names.

Out-of-order processing (OOOP) at recursive resolvers avoids head-of-line blocking in pipelines. OOOP is defined and explicitly allowed by RFC-5966 [2]. Without OOOP, queries to even a small percentage of distant servers will stall a strictly-ordered queue, unnecessarily delaying all subsequent queries. Without connections, concurrent UDP queries are naturally independent and all major DNS servers process them concurrently. We know of no DNS server today that supports out-of-order processing of TCP queries; `BIND` and `Unbound` instead resolve each query for a TCP connection before considering the next.

Two protocol optimizations can speed re-opening of closed connections. *TCP fast open* [4] allows data to be exchanged in the three-way handshake packets, saving one round-trip time (RTT) in the TCP handshake. With *TLS resumption* [14], a sever gives the client all state needed to securely re-create a TLS connection, saving one RTT in TLS setup.

We describe strategies for gradual deployment elsewhere [17].

4. T-DNS PERFORMANCE EVALUATION

The main contribution of our work is a careful performance study of T-DNS to show that TCP and TLS have reasonable cost. We show that connection reuse works well, that servers require a reasonable amount of state, and that clients have only modest latency improvements.

Connection reuse and server state: Connection caching poses a fundamental trade-off: clients prefer long-lived connections (they have resources and hate latency), but servers prefer short-lived connections (they share resources over may clients). We examine this trade-off, varying connection timeout to measure the *connection hit fraction*, how often an existing connection can be reused without setup, and *concurrent connections*, how many connections are active on a server at any time. We recommend timeouts of 60 s for recursive resolvers and 20 s for authoritative servers based on analysis of traces and conservative choices. Trace analysis with these values shows 85% connection hit rates, and Figure 1 shows the number of concurrent connections. If we assume 150 kB memory per connection, the 75%iles suggest the Level 3 recursive resolver requires 3.6 GB RAM, and the B-Root authoritative resolver requires about 7.4 GB. These values are well within current, commodity server hardware.

End-to-end Latency: For clients, the primary cost of T-DNS is the additional latency due to connection setup. We experimentally evaluate the stub-to-recursive and recursive-to-authoritative latency with TCP and TLS respectively in prototyped implementations. These results show that connection setup latency is dominated by protocol RTT requirements, and that connection reuse can completely remove setup latency in the 85% of connection hits.

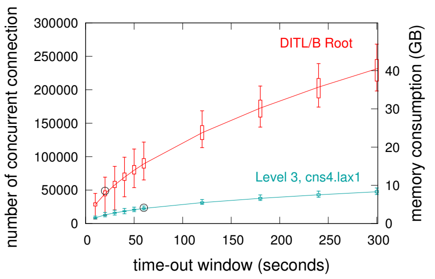

Figure 1: Median and quartiles of numbers concurrent connections. Datasets: Level 3/cns4.lax1 and B-Root

We model the expected (average) *end-to-end* latency for DNS users. We see that *use of TCP and TLS to the local resolver adds moderate latency*: TLS is only 9% slower with UDP upstream. Second, we see that use of connections between recursive and authoritative is very expensive: with TLS stub-to-recursive, adding TCP to the authoritative is 22% slower and adding TLS to the authoritative is more than 170% slower. This cost follows because a single stub-to-recursive query can lead to multiple recursive-to-authoritative queries, at large RTTs with a lower connection-hit fraction.

5. REFERENCES

[1] Anonymous. The collateral damage of internet censorship by DNS injection. *SIGCOMM CCR*, June 2012.

[2] R. Bellis. DNS Transport over TCP - Implementation Requirements. RFC 5966, Aug. 2010.

[3] S. Bortzmeyer. DNS privacy problem statement. Internet draft, Dec. 2013.

[4] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain. TCP fast open, Dec. 2011. Work in progress (Internet draft draft-cheng-tcpm-fastopen-02).

[5] J. Damas, M. Graff, and P. Vixie. Extension mechanisms for DNS (EDNS(0)). RFC 6891, Apr. 2013.

[6] W. Eddy. TCP SYN flooding attacks and common mitigations. RFC 4987, Aug. 2007.

[7] G. Greenwald. NSA collecting phone records of millions of Verizon customers daily. *The Guardian*, June 2013.

[8] A. Herzberg and H. Shulmanz. Fragmentation considered poisonous. IEEE-CNS, Oct. 2013.

[9] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, and D. Wessels. Starting TLS over DNS. Internet draft, Jan. 2014.

[10] ICANN. Root server attack on 6 February 2007. Technical report, Mar. 2007.

[11] C. A. Kent and J. C. Mogul. Fragmentation considered harmful. In *SIGCOMM*, pages 390–401, Aug. 1987.

[12] W. Meng, R. Duan, and W. Lee. DNS Changer remediation study. Talk at M3AAWG 27th, Feb. 2013.

[13] C. Metz. Comcast trials (domain helper service) DNS hijacker. The Register, July 2009.

[14] J. Salowey, H. Zhou, P. Eronen, and H. Tschofenig. Transport Layer Security (TLS) Session Resumption without Server-Side State. RFC 5077, Jan. 2008.

[15] W. Simpson. TCP cookie transactions, Jan. 2011.

[16] R. Vaughn and G. Evron. DNS amplification attacks. `http://isotf.org/news/DNS-Amplification-Attacks.pdf`, Mar. 2006.

[17] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, and N. Somaiya. T-DNS: Connection-oriented DNS to improve privacy and security (extended). Technical Report ISI-TR-2014-693, USC/ISI, June 2014.