T-DNS: Connection Oriented DNS to Improve Privacy and Security

John Heidemann
joint work with Liang Zhu, Zi Hu,
Duane Wessels, Allison Malkin, Nikita Somaiya

1 USC/ISL, 2 Verisign Labs
10 May 2014

---

 DNS Basics

since 1987 (RFC-1034)
DNS is simple request-response:

client: A www.example.com?

server: 192.0.2.1

perfect for UDP
(TCP supported too, but as fallback and zone transfers)

---

 Fear of DNS over TCP

• TCP is horribly slow: *bad client latency*
• TCP => server state: *server memory explodes*

community consensus: *orthodoxy* dogma

*don’t use TCP*, UDP’s constraints are OK

* except for fallback and zone transfers

---

 Our Contributions

• analysis: *don’t fear connections for DNS*
  – client latency: only modestly more
  – server memory: well within current hardware
• implementation choices to get here
• small protocol addition: TLS upgrade

=> T-DNS: *DNS over TCP+TLS*

---

 T-DNS: TCP and TLS Connections

• introduction
• **why**
  • how
  • at minimal cost
  • better than alternatives
  • next steps
Why T-DNS

- protecting privacy
  - connections → encryption → privacy
- denying DoS (Denial of Service)
  - connections → spoof-proof → no amplification attacks
- leaving limits
  - connections → UDP limits don’t drive policies

Protecting Privacy

- principle: all traffic should be private (⇒ encrypted)
- rise of public DNS means many can eavesdrop
  - Google Public DNS, OpenDNS, others
  - traffic over WAN should be private!
- individuals avoiding transparent proxies
  - multiple ISPs intercept DNS to add ads
- DNS is more than addresses
  - anti-privacy (DNSBL), embedded user IDs (Facebook, etc.)
  - DNSBL’s open check unique IP address of every incoming mail server over the WAN
  - even on LAN (where destinations are visible), should protect other content

Denying DoS

- problem: DNS attacks others
  - DNS amplification attacks
    - a growth industry in 2013: >100Gbps attacks
  - problem: DoS on DNS servers
    - work-around: massive over-capacity
  - solution: TCP
    - well understood anti-DoS methods:
      - 3-way handshake precludes spoofing
      - TCP cookies shift state to client for non-spoofed

Leaving Limits

- for >25 years, policy decisions forced by UDP packet sizes
  - number of root servers: all fit in 512B
  - DNSsec: required EDNS for >512B
  - crypto algo and key sizes: pkt size limited
  - key rollover: temporary 2x size
- partial fix: EDNS0 deployment (10+ years, since 1999)
- what uses already discarded as too big? ⇒ enough already!

Doesn’t DNSsec already “Secure DNS”?

A: yes, but...

- DNSsec is about query integrity
  - that is: if you are told X, is X true?
  - it signs answers; signatures prove X is true
- DNS does nothing for problems
  - everything sent in the clear: no privacy
  - nothing about DoS
  - large signatures stress UDP size limits

⇒ need DNSsec’s integrity and T-DNS’ privacy

T-DNS: TCP and TLS Connections

- introduction
- why
  - how
  - at minimal cost
  - better than alternatives
  - next steps
(Review) Our Contributions

- analysis: **don’t fear connections for DNS**
  - client latency: only modestly more
  - server memory: well within current hardware
- implementation choices to get here
- small protocol addition: TLS upgrade

(Review) Our Contributions

3. analysis: **don’t fear connections for DNS**
   - client latency: only modestly more
   - server memory: well within current hardware
2. implementation choices to get here
1. **small protocol addition: TLS upgrade**
   (going in reverse order)

Protocol Changes: Goals

- minimize change
- reuse existing approaches  
  (as boring as possible)
- follow IETF patterns

Protocol Changes: Goals

- minimize change
- reuse existing approaches  
  (as boring as possible)
- follow IETF patterns

  **implications:**
  - reuse TLS: Transport Layer Security
  - add a STARTTLS-like “upgrade”
  - innovation: careful implementation

SMTP before STARTTLS

| C & S: open TCP connection |
| S: 220 mail.example.com SMTP service ready |
| C: EHLO mail.example.com |
| S: 250 mail.example.org bi, extensions are: -ABRITIME -STARTTLS DSN |
| C: MAIL FROM<<sender@mail.example.com>> |
| S: 250 2.1.0 <<sender@mail.example.com>>.. Sender OK |
| C: RCPT TO<<destination@example.com>> |
| S: 250 2.1.5 <destination@example.com>> |
| C: <send mail contents>> |

problem: cleartext mail is snoop-able  
(fix: TLS)

SMTP with STARTTLS

| C & S: open TCP connection |
| S: 220 mail.example.com SMTP service ready |
| C: EHLO mail.example.com |
| S: 250 mail.example.org bi, extensions are: -ABRITIME -STARTTLS DSN |
| C: STARTTLS |
| S: 220 Go ahead |
| C & S: <negotiate a TLS session with a new session key, in binary> |
| C: EHLO mail.example.com |
| S: 250 mail.example.org hello, extensions are: -ABRITIME DSN |
| C: MAIL FROM<<sender@mail.example.com>> |
| S: 250 2.1.0 <<sender@mail.example.com>>.. Sender OK |
| C: RCPT TO<<destination@example.com>> |
| S: 250 2.1.5 <destination@example.com>> |
| C: <send mail contents>> |

this example: SMTP;  
idea used for IMAP, POP3, FTP, XMPP, LDAP, NNTP…

prologue: in clear  
(no privacy here)

transition to TLS  
contents now private
Our STARTTLS for DNS
(in draft-hzwhm-start-tls-for-dns-01)

C & S: open TCP connection
prologue
transition to TLS
C: QNAME="STARTTLS", QCLASS=CH, QTYPE=.TXT
with the new TID bit set in EDNS options
S: RCODE=0, TXT="STARTTLS", with the TO bit set
C & S: <negotiate a TLS session, get new session key, in
binary>
contests now private
C: <send actual query>
S: <reply to actual query>
pros: no new port (from IANA, or in firewalls)
cons: extra RTT; middleboxes may not like encrypted tcf

Careful Implementation Choices

- problem: no tuning of DNS TCP for queries
  (until now!)
- connection reuse (or restart)
  – persistent connections
  – TCP fast open
  – TLS resumption
- query pipelining
- out-of-order processing

Latency in DNS/TLS

C & S: open TCP connection
TCP 3wh: +1 RTT
C: QNAME="STARTTLS", QCLASS=CH, QTYPE=.TXT
with the new TID bit set in EDNS options
S: RCODE=0, TXT="STARTTLS", with the TO bit set
C & S: <negotiate a TLS session with a new session key, in
binary>
TLS handshake: +2 RTTs
C: <send actual query>
S: <reply to actual query>
query: 1 RTT

Connection Reuse

- basic idea:
  reuse connection -> no setup cost

- secondary idea:
  if must close, client keeps state to restart quickly

(Review) Our Contributions

3. analysis: don’t fear the DNS connection
   – client latency: only modestly more
   – server memory: well within current hardware
2. implementation choices to get here
   1. small protocol addition: TLS upgrade
   (going in reverse order)

Connection Reuse

- basic idea:
  reuse connection -> no setup cost
  – persistent connections (in client and server)
- secondary idea:
  if must close, client keeps state to restart quickly
  – TCP fast open: client has cookie to send data in 3wh
    • draft-ietf-tcpm-fastopen-08: in Linux-3.6, default 3.13
  – TLS resumption (RFC-5077): client keeps
    • RFC-5077: in OpenSSL and GnuTLS
Query Pipelining
send several queries immediately (not stop-and-wait)

before pipelining

with pipelining

pipelining matters:
62% of web has 4+ domain names (even concurrent)

Out-of-Order Processing
process queries on same connection in parallel

in-order (only)

out-of-order processing

queries run in parallel

reply as soon as possible (maybe reorder)

T-DNS: TCP and TLS Connections

• introduction
• why
• how
• at minimal cost
• better than alternatives
• next steps

(Review) Our Contributions

3. analysis: don’t fear connections for DNS
   – client latency: only modestly more
   – server memory: well within current hardware
2. implementation choices to get here
1. small protocol addition: TLS upgrade

(going in reverse order)

(Review) Our Contributions

3. analysis: don’t fear connections for DNS
   – client latency: only modestly more
   – server memory: well within current hardware

questions:
  a. connection reuse: hit rate? memory?
  b. CPU cost?
  c. latency:
     i. stub-recursive?
     ii. recursive-authoritative?
     iii. end-to-end?

Connection Reuse Helps? (YES!)

what fraction of queries find open TCP connections?

method: replay 3 traces: recursive (DNSchanger, Level3) and authoritative (R-Root)

DnsChange-1-nt: Level 3, 91.102.164.138 (active)
DnsChange-2-nt: Level 3, 93.184.216.138 (active)
DnsChange-3-nt: Level 3, 93.184.216.138 (active)

120s timeout => >40% connection reuse (reuse is effective?)

set proposal 30s/80s (conservative) => still >85% connection reuse

conclusion: connection reuse is often helpful
**Cost of Connection Reuse? (ok!)**

- How many connections?
- How much memory?
- Method: replay same 3 traces (then we saw 2 biggest)
- Experimental estimate of memory: 300kB/conn (very conservative)

Conclusion: connection reuse is often helpful and it’s not too costly (easy to add server parallelism if needed)

**Latency: CPU Cost**

- We used micro-benchmarks to study CPU cost
- \( \text{CPU cost} \)
- OpenSSL vs. GeaTLS:
  - TCP handshake processing: 0.15 ms vs. 0.12 ms
  - TCP packet handling: 25.8 ms vs. 8 ms
  - TLS connection establishment:
    - key exchange: 25.8 ms vs. 8 ms
    - CA validation: 12.3 ms vs. 1.5 ms
    - TLS connection resumption:
      - DNS resolution (from cache): 0.1 ms vs. 0.5 ms
- TLS setup is noticeable, but RTT (40-100+ ms) more important.

**Latency: Stub to Recursive**

- TCP and TLS vs. UDP:
  - Effects of implementation choices:
    - With short RTT (1 ms)
    - Method: live experiments of random 140 names from Alexa top 1000, stub-recursive RTT: 1 ms (graph shows median and quartiles)

**Latency: Stub to Recursive**

- TCP and TLS vs. UDP:
  - Effects of implementation choices:
    - With long RTT (~35 ms)
    - Method: live experiments of random 140 names, each reaped 10x, recursive-authorized RTT: 35 ms (graph shows median and quartiles)

**Latency: Recursive to Authoritative**

- TCP and TLS vs. UDP:
  - Effects of implementation choices:
    - With long RTT (~35 ms)
    - Method: live experiments of random 140 names, each reaped 10x, recursive-authorized RTT: 35 ms (graph shows median and quartiles)

**Latency: Recursive to Authoritative**

- TCP and TLS vs. UDP:
  - Effects of implementation choices:
    - With long RTT (~35 ms)
    - Method: live experiments of random 140 names, each reaped 10x, recursive-authorized RTT: 35 ms (graph shows median and quartiles)
End-to-End Latency: Methodology

- controlled experiments are hard
  - variable stub query timing
  - caching at recursive resolver
  - different RTTs (many stubs and authoritative)

- approach: model expected latency
  - i.e., just averages
  - median connection reuse from trace replay
  - other parameters from experiments

End-to-End Latency: Results

T-DNS: TCP and TLS Connections

- introduction
- why
- how
- at minimal cost
  - better than alternatives
- next steps

Alternatives

- for improving privacy
  - DNScurve/DNScrypt: some neat optimizations to reduce RTTs, but new and fixed stack
  - DNS over DTLS: adds back UDP limits but still stuck with most TLS RTTs
- for anti-DoS
  - on others: rate limiting
  - for relaxing limits:
    - seeming alternative: live within UDP limits

T-DNS Next Steps

- more information:
  - tech report ISI-TR-2014-688 (www.isi.edu/~johnh/PAPERS/Zhu14a/)
  - internet-draft: draft-hzhwms-start-tls-for-dns-01
- code:
  - client, client & server proxies, unbound patch
  - http://www.isi.edu/ant/software/
- do you want DNS privacy? share feedback?
  - johnh@isi.edu