The Importance of Being an Earnest stub

Challenges and solution for the versatile stub

Willem Toorop
13 May 2017
OARC 26 (Madrid)
From the ground-up security

- Every “secure” connection is preceded by a DNS lookup
- The stub does the lookup at the request of the application
  The recursive resolver does all the heavy lifting
From the ground-up security

- DNSSEC protects against cache poisoning
From the ground-up security

- DNSSEC protects against cache poisoning
- But not against resolver hijacking (i.e. ARP or DHCP hijacking or routing tricks)
From the ground-up security

- DNSSEC protects against cache poisoning
- But not against resolver hijacking
- One possibility: DNSSEC on the stub
From the ground-up security/privacy

- DNSSEC protects against cache poisoning
- But not against resolver hijacking
- Another possibility: DNS over TLS
From the ground-up, security/privacy

- TLS hijacking? **IS THAT POSSIBLE?!**

---

![Diagram of DNS over TLS interception](image)

**Fig. 2: Detecting Interception**—We quantify HTTPS interception at three major Internet services. We estimate that 5–10% of connections are intercepted.

<table>
<thead>
<tr>
<th>Vantage Point</th>
<th>% HTTPS Connections Intercepted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Interception</td>
</tr>
<tr>
<td>Cloudflare</td>
<td>88.6%</td>
</tr>
<tr>
<td>Firefox</td>
<td>96.0%</td>
</tr>
<tr>
<td>E-commerce</td>
<td>92.9%</td>
</tr>
</tbody>
</table>

Applies to DNS over TLS too.
From the ground-up security/privacy

- Strengthen TLS security with the stub: DANE
  (DNS-based Authentication of Named Entities)

- Also signalling system for TLS support
  (For application without user interaction)
From the ground-up security/privacy

Authenticate DNS-over-TLS with DANE?

- Bootstrap the TLSA lookup with regular DNS?
From the ground-up security/privacy

Authenticate DNS-over-TLS with DANE?

- Bootstrap the TLSA lookup with regular DNS?
  - Chicken and Egg problem
From the ground-up security/privacy

- Bootstrap the TLSA lookup with regular DNS?
- Have the TLSA record + the complete DNSSEC authentication chain embedded in a TLS extension

From the ground-up security/privacy

- Bootstrap the TLSA lookup via a stub.
- Have the TLSA record + the complete DNSSEC authentication chain embedded in a TLS extension.


Emergency: TLS DNSSEC authentication chain extension must be obligatory, to prevent the “Too many CA’s” problem.
From the ground-up security/privacy

DNSSEC Availability

- DNS Privacy status
  - Clear text DNS
  - Private DNS
  - Authenticated Private DNS

• The stub is close to the application
  Inform status of DNSSEC and DNS Privacy
From the ground-up security/privacy

- Enhanced privacy by round-robinning upstreams
From the ground-up security/privacy

- Requirements for the versatile stub

<table>
<thead>
<tr>
<th>Feature</th>
<th>DNSSEC</th>
<th>DNS over TLS</th>
<th>Non address lookups</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross the first DNSSEC mile</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>From the ground up Privacy</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strengthened TLS authentication (DANE)</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strengthened opportunistic TLS (DANE)</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide status of DNSSEC &amp; DNS over TLS</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Willem Toorop (NLnet Labs)
## From the ground-up security/privacy

### Requirements for the versatile stub

<table>
<thead>
<tr>
<th>Cross the first DNSSEC mile</th>
<th>DNSSEC</th>
<th>DNS over TLS</th>
<th>Non address lookups</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>From the ground up Privacy</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strengthened TLS authentication (DANE)</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Strengthened opportunistic TLS (DANE)</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Provide status of DNSSEC &amp; DNS over TLS</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
**DNSSEC Roadblocks**

- Resolving DNSSEC *(to cross the first mile)* needs DNSSEC Aware recursive resolver
• Resolving DNSSEC *(to cross the first mile)* needs DNSSEC Aware recursive resolver

DNSSEC Roadblocks

- Resolving DNSSEC requires a DNSSEC Aware recursive resolver.
DNSSEC Roadblocks

• DNSSEC Roadblock Avoidance  https://tools.ietf.org/html/rfc8027
• IPv6 Address Synthesis Prefix Discovery  
  +DNS64 capability  
DNSSEC Roadblocks

  +DNS64 capability  https://tools.ietf.org/html/rfc6147

Willem Toorop (NLnet Labs)
DNSSEC Roadblocks

Root KSK Rollover

- DNSSEC validating stubs must do RFC5011
DNSSEC Roadblocks

- DNSSEC stubs must do RFC5011
  - Root KSK Rollover
  - In-band RFC5011 tracking with DNSSEC auth chain
  - TLS extension

In-band RFC5011 tracking with DNSSEC auth chain TLS extension
DNSSEC Roadblocks

- DNSSEC validating stubs must do RFC5011
- A stub library for DANE has no system config +bootstrap DNSSEC capability: https://tools.ietf.org/html/rfc7958
- A stub library for DANE runs with user's privileges

Root
KSK
Rollover

Willem Toorop (NLnet Labs)
## DNSSEC Roadblocks

### DNSSEC stubs capability requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNSSEC validation</td>
<td>(various)</td>
</tr>
<tr>
<td><strong>DNSSEC Roadblock Avoidance</strong></td>
<td>RFC8027</td>
</tr>
<tr>
<td>IPv6 Prefix Discovery</td>
<td>RFC7050</td>
</tr>
<tr>
<td>IPv6 Address Synthesis</td>
<td>RFC6147</td>
</tr>
<tr>
<td>Automated Trust Anchor Updates</td>
<td>RFC5011</td>
</tr>
<tr>
<td>Automated Initial Trust Anchor retrieval</td>
<td>RFC7958</td>
</tr>
</tbody>
</table>
# From the ground-up security/privacy

- Requirements for the versatile stub

<table>
<thead>
<tr>
<th>Cross the first DNSSEC mile</th>
<th>DNSSEC</th>
<th>DNS over TLS</th>
<th>Non address lookups</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>From the ground up Privacy</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strengthened TLS authentication <em>(DANE)</em></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strengthened opportunistic TLS <em>(DANE)</em></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide status of DNSSEC &amp; DNS over TLS</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Requirements for DNS-over-TLS

- TCP fastopen *(optional)*
- Connection reuse
- EDNS0 keepalive
- EDNS0 padding
Requirements for DNS-over-TLS

- Connection reuse
  
- Pipe-lining of queries
  
(Q/R, Q/R, Q/R)

(Q, Q, Q, R, R, R)
Requirements for DNS-over-TLS

- Connection reuse
- Pipe-lining of queries
- Process Out-Of-Order-Responses

\[(Q/R, Q/R, Q/R)\]
\[(Q,Q,Q,R,R,R)\]
\[(Q_1, Q_2, R_2, R_1)\]
Requirements for DNS-over-TLS

- Strict or Opportunistic usage profiles?

  1) Authenticated Private DNS
  2) Private DNS
  3) Clear text DNS
Requirements for DNS-over-TLS

- Strict or Opportunistic usage profiles?

RFC7858 (DNS-over-TLS) defined direct SPKI authentication only

3) Clear text DNS
Requirements for DNS-over-TLS

- Regular PKIX authentication
  *(bootstrap address lookup with regular DNS(SEC))**
Requirements for DNS-over-TLS

- Regular PKIX authentication
- Authenticate with DANE
  (strictly opportunistic with TLSA signalling)
Requirements for DNS-over-TLS

- Regular PKIX authentication
- Authenticate with DANE
- DNSSEC authentication chain TLS extension
## Requirements for DNS Privacy

<table>
<thead>
<tr>
<th>Requirement</th>
<th>RFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS-over-TLS</td>
<td>RFC7858</td>
</tr>
<tr>
<td>Reuse / Pipelining / OOOR</td>
<td>RFC7766</td>
</tr>
<tr>
<td>TCP Fastopen</td>
<td>RFC7413</td>
</tr>
<tr>
<td>ENDS0 keepalive</td>
<td>RFC7828</td>
</tr>
<tr>
<td>ENDS0 padding</td>
<td>RFC7830</td>
</tr>
<tr>
<td>PKIX support for authentication</td>
<td>(various)</td>
</tr>
<tr>
<td>DNSSEC support</td>
<td>(various)</td>
</tr>
<tr>
<td>(for address lookup and authentication)</td>
<td></td>
</tr>
</tbody>
</table>
From the ground-up security/privacy

- **Requirements for the versatile stub**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>DNSSEC</th>
<th>DNS over TLS</th>
<th>Non address lookups</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross the first DNSSEC mile</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>From the ground up Privacy</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strengthened TLS authentication (DANE)</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strengthened opportunistic TLS (DANE)</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide status of DNSSEC &amp; DNS over TLS</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Non address lookups - Application Interface

getaddrinfo() and getnameinfo()
(POSIX standard extended by RFC3493 for IPv6)
Non address lookups - Application Interface

getaddrinfo() and getnameinfo()
(POSIX standard extended by RFC3493 for IPv6)

Talk to upstreams directly with a library:
- libresolv, libval, ldns, libunbound, libgetdns

Learn upstreams from OS
- /etc/resolv.conf, NetworkManager, registry...
Non address lookups - Application Interface

Applications using `getaddrinfo()` API will not get the versatile stub features (first DNSSEC mile coverage, DNS privacy)

Talk to upstreams directly with a library:
- `libresolv`, `libval`, `ldns`, `libunbound`, `libgetdns`

Learn upstreams from OS
- `/etc/resolv.conf`, `NetworkManager`, `registry`...
Non address lookups - Application Interface

Stub server listening on 127.0.0.1:53

- `getaddrinfo()` and `getnameinfo()`
  use system stub which uses stub server
Non address lookups - Application Interface

getaddrinfo() and getnameinfo() use systemd-resolved via nsswitch module

- Stub server listening on 127.0.0.53:53

```
systemd-resolved.service
```

```
systemd-resolved
```
Non address lookups - Application Interface

Talk to stub server via a library:

- *libresolv*, *libval*, *ldns*, *libunbound*, *libgetdns*

**systemd-resolved.service**

127.0.0.53:53

**Dnssec-Trigger**

**Stubby**

**getdns**

**Dnsmasq**
Non address lookups - Application Interface

Talk to stub server via a library:
- `libresolv`, `libval`, `ldns`, `libunbound`, `libgetdns`

.stub

Stubby

Dnssec Trigger

Dnsmasq

systemd-resolved

127.0.0.1:53:53

systemd-resolved.service
Non address lookups - Application Interface

Talk to stub server via the dbus API
- https://www.freedesktop.org/wiki/Software/systemd/resolved/

systemd-resolved.service
systemd-resolved
The Importance of Being an Earnest stub